Hello, Authelia

Authelia

Authelia Web 2FASSO

Authelia

 VaultWarden Homepage SSO

 1Panel SSO Logto Logto PostgreSQL SQLite Authelia

1Panel Authelia

 1Panel 1Panel 1Panel PR GitHub lite lite

 data configuration.yml Authelia

```yaml
# configuration.yml
server:
  address: 'tcp://:9091'

authentication_backend:
  file:
    path: '/config/users_database.yml'

totp:
  disable: false
  issuer: 'authelia.com' # 

identity_validation:
  reset_password:
    jwt_algorithm: 'HS512'

identity_providers:
  oidc:
    jwks:
      - key_id: 'example' # 
        algorithm: 'RS256'
        use: 'sig'
        key: { { secret "/config/secrets/oidc/jwks/rsa.2048.key" | mindent 10 "|" | msquote } }
```

```yaml

session:
  cookies:
    - name: 'authelia_session' # 随便□个名字
      domain: 'example.com' # □为你的域名
      authelia_url: 'https://authelia.example.com' #
      expiration: '1 hour'
      inactivity: '5 minutes'
      remember_me: '1 week'

access_control:
  default_policy: 'deny'
  rules:
    - domain: 'public.example.com' # □为你的域名
      policy: 'bypass' #
    - domain: 'private.example.com' # □为你的域名
      policy: 'one_factor' #
    - domain: 'secure.example.com' # □为你的域名
      policy: 'two_factor' #

regulation:
  max_retries: 3
  find_time: '2 minutes'
  ban_time: '5 minutes'

storage:
  local:
    path: '/config/db.sqlite3'

notifier:
  filesystem:
    filename: '/config/notification.txt'
  # smtp: # 要用 smtp 就把上面的注释掉
  #   username: 'test'
  #   password: 'password'
  #   address: 'smtp://mail.example.com:25'
  #   sender: 'admin@example.com'
```

接着要修改保存在文件中的各个机密：

- `STORAGE_ENCRYPTION`：应为不低于 20 位的随机字符串
- `SESSION_SECRET`：应为不低于 64 位，且仅包含大小写字母与数字的随机字符串
- `JWT_SECRET`：应为不低于 64 位，且仅包含大小写字母与数字的随机字符串
- `HMAC_SECRET`：应为不低于 64 位，且仅包含大小写字母与数字的随机字符串
- `oidc/jwks/rsa.2048.key` 与 `oidc/jwks/rsa.2048.key.pub`：应为使用 RSA 方法生成的、位数不低于 2048 的一对公私钥

所有机密均可通过 Authelia 提供的生成器生成，在容器终端中执行命令即可：

- 随机字符串：

  ```
  authelia crypto rand --length 64 --charset alphanumeric
  ```

- RSA 密钥对：

```
authelia crypto pair rsa generate --directory /config/secrets/oidc/jwks --file.private-key rsa.2048.key --file.public-key rsa.2048.key.pub
```

最后修改 users_database.yml，添加上自己的账户，其中密码需要在容器终端中执行命令：

```
authelia crypto hash generate argon2 --password 'password'
```

或者直接生成随机密码：

```
authelia crypto hash generate argon2 --random --random.length 64 --random.charset alphanumeric
```

然后将 `Digest` 的内容放到 `password` 字段中，Authelia 就算是部署完成了。

⊠⊠⊠⊠⊠⊠⊠⊠

当我们部署的站点较多的时候，管理大量非公开站点的登录信息就成了一个问题。如果只是单人使用，或许还可以靠密码管理工具来简单管理；但使用人数一多，光是帮每个使用者注册账户就够喝一壶了。Authelia 这类单点登录工具便为此而生。下面我以官方文档中没有示例的 Alist 接入为例，演示如何为一步步为一个支持 OIDC 登录，但是没有 Authelia 教程的站点接入 Authelia。

阅读 Alist 文档，可以找到 Alist 单点登录的 callback 链接：

```
https://your.alist.domain/api/auth/sso_callback\?method=sso_get_token
https://your.alist.domain/api/auth/sso_callback\?method=get_sso_id
```

于是可以根据这个链接使用 Authelia 设置单点登录。首先在 Authelia 的 configuration.yml 中添加如下设置：

```
# configuration.yml
identity_providers:
  oidc:
    clients:
      - client_id: 'alist'
        client_name: 'Alist'
        client_secret: '$pbkdf2-
sha512$310000$c8p78n7pUMln0jzvd4aK4Q$JNRBzwAo0ek5qKn50cFzzvE9RXV88h1wJn5KGiHrD0YKt
nCb2CJPOsKaPK0hjf.9yHxzQGZziziccp6Yng' # 示例密钥哈希值，对应客户端示例密钥为 `insecure_secret`
        public: false
        authorization_policy: 'one_factor'
        redirect_uris:
          - 'https://alist.example.com/api/auth/sso_callback?method=sso_get_token'
          - 'https://alist.example.com/api/auth/sso_callback?method=get_sso_id'
        scopes:
          - 'openid'
          - 'profile'
        userinfo_signed_response_alg: 'none'
        token_endpoint_auth_method: 'client_secret_post' # Alist 使用 POST 进行单点登录，不设置会报错，提示使用该设置。
```

然后，在 Alist 的单点登录设置页面选择平台为 OIDC，设置如下选项：

- ⊠点登录客户端 **ID**: `alist`
- ⊠点登录客户端机密: `insecure_secret`
- **Sso oidc** 用户名键: `preferred_username`
- ⊠点登录端点名称: `https://authelia.example.com`

重启 Authelia，退出 Alist 登录，现在就可以通过单点登录登录到 Alist 了。

> [!info] 一些设置项的来历

- **Sso oidc** 用户名键: 来自 Authelia 的文档，一部分集成指南中使用了类似的配置。
- 单点登录端点名称: 先随意设置了一个名称，报错信息显示其访问了 .well_know-known 路径，所以使用 Authelia 部署根路径。

站点外挂登录系统

还有一些站点根本没有用户登录这个功能，但有时候我们又不想对外公开这个站点，因此我们需要为网站外挂一个登录系统。Authelia 便可以很好地承担这项任务，对于 1Panel 用户来说，只需要在网站的配置文件中添加少量代码，即可为网站接入 Authelia 的登录系统。

1Panel 的网站管理使用的是基于 Nginx 的 OpenRestry，因此只要是基于 Nginx 的反向代理网站管理系统，都可以参考接下来的配置方法。

打开 1Panel 中需要外挂登录的网站配置页面，找到反向代理一栏，添加一个新的反向代理，修改源文如下：

```
location /internal/authelia/authz {
    proxy_pass http://127.0.0.1:9091/api/authz/auth-request; # 取决于 Authelia 部署的位置与端口号
    proxy_set_header Host $host;
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_set_header X-Forwarded-Host $http_host;
    proxy_set_header X-Forwarded-URI $request_uri;
    proxy_set_header X-Forwarded-Ssl on;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header X-Original-Method $request_method;
    proxy_set_header X-Original-URL $scheme://$http_host$request_uri;
    proxy_set_header X-Forwarded-For $remote_addr;
    proxy_set_header Content-Length "";
    proxy_set_header Connection "";
    proxy_pass_request_body off;
    proxy_next_upstream error timeout invalid_header http_500 http_502 http_503;
    proxy_redirect http:// $scheme://;
    proxy_http_version 1.1;
    proxy_cache_bypass $cookie_session;
    proxy_no_cache $cookie_session;
}
```

然后修改默认的反向代理源文，添加一段内容：

```
location / {
    auth_request /internal/authelia/authz;
    auth_request_set $user $upstream_http_remote_user;
    auth_request_set $groups $upstream_http_remote_groups;
    auth_request_set $name $upstream_http_remote_name;
    auth_request_set $email $upstream_http_remote_email;
    proxy_set_header Remote-User $user;
    proxy_set_header Remote-Groups $groups;
    proxy_set_header Remote-Name $name;
    proxy_set_header Remote-Email $email;
    auth_request_set $redirection_url $upstream_http_location;
    error_page 401 =302 $redirection_url;

```

```
    proxy_pass http://127.0.0.1:23423;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    # ...
}
```

□□□□□□□□